

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

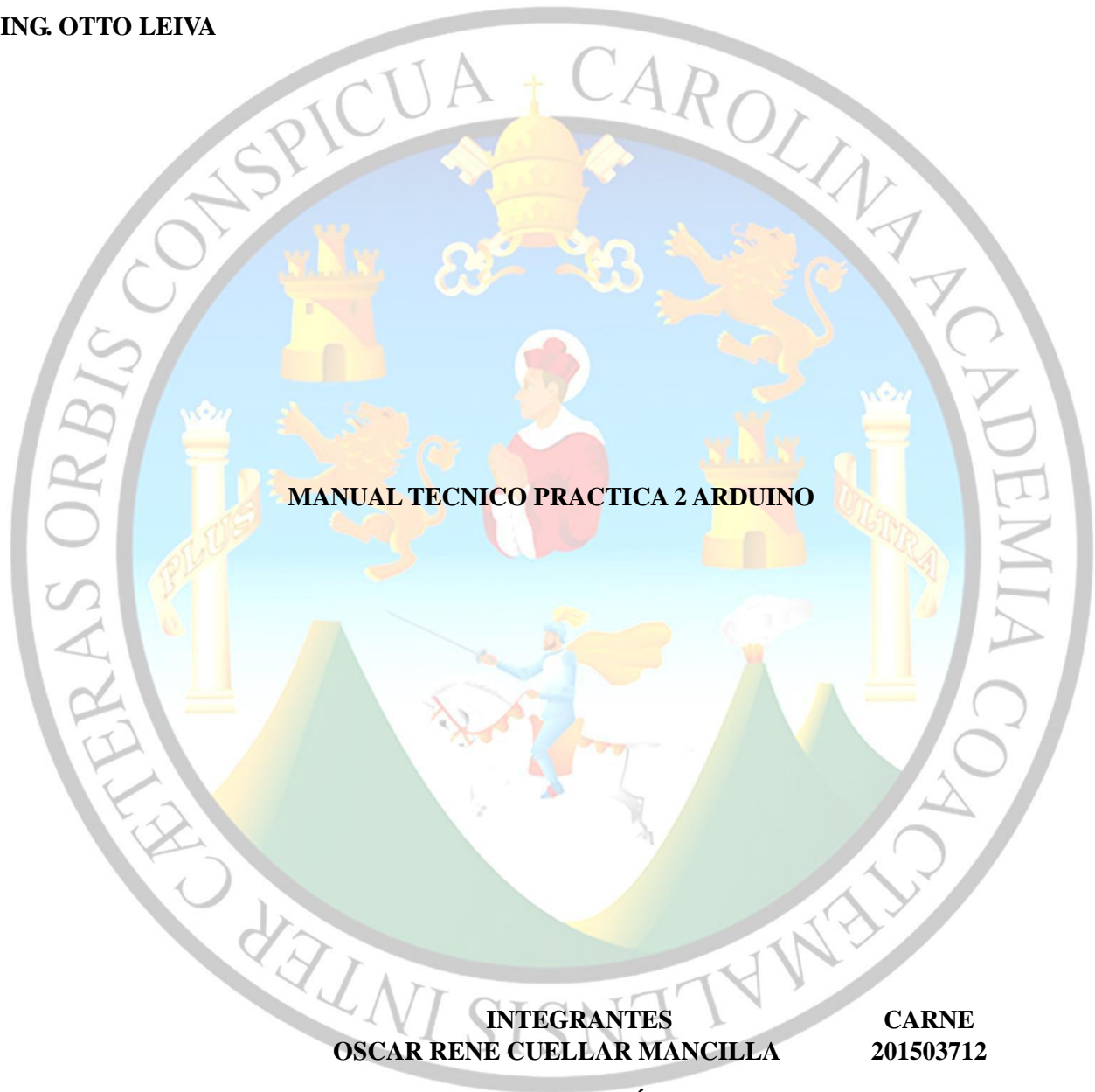
FACULTAD DE INGENIERIA

CIENCIAS Y SISTEMAS

ARQUITECTURA DE COMPUTADORES 1

AUX. JORGE GUTIERREZ

ING. OTTO LEIVA



INTEGRANTES

OSCAR RENE CUELLAR MANCILLA

**CARNE
201503712**

HAYRTON OMAR IXPATÁ COLOCH

201313875

NERY GONZALO GALVEZ GÓMEZ

201403525

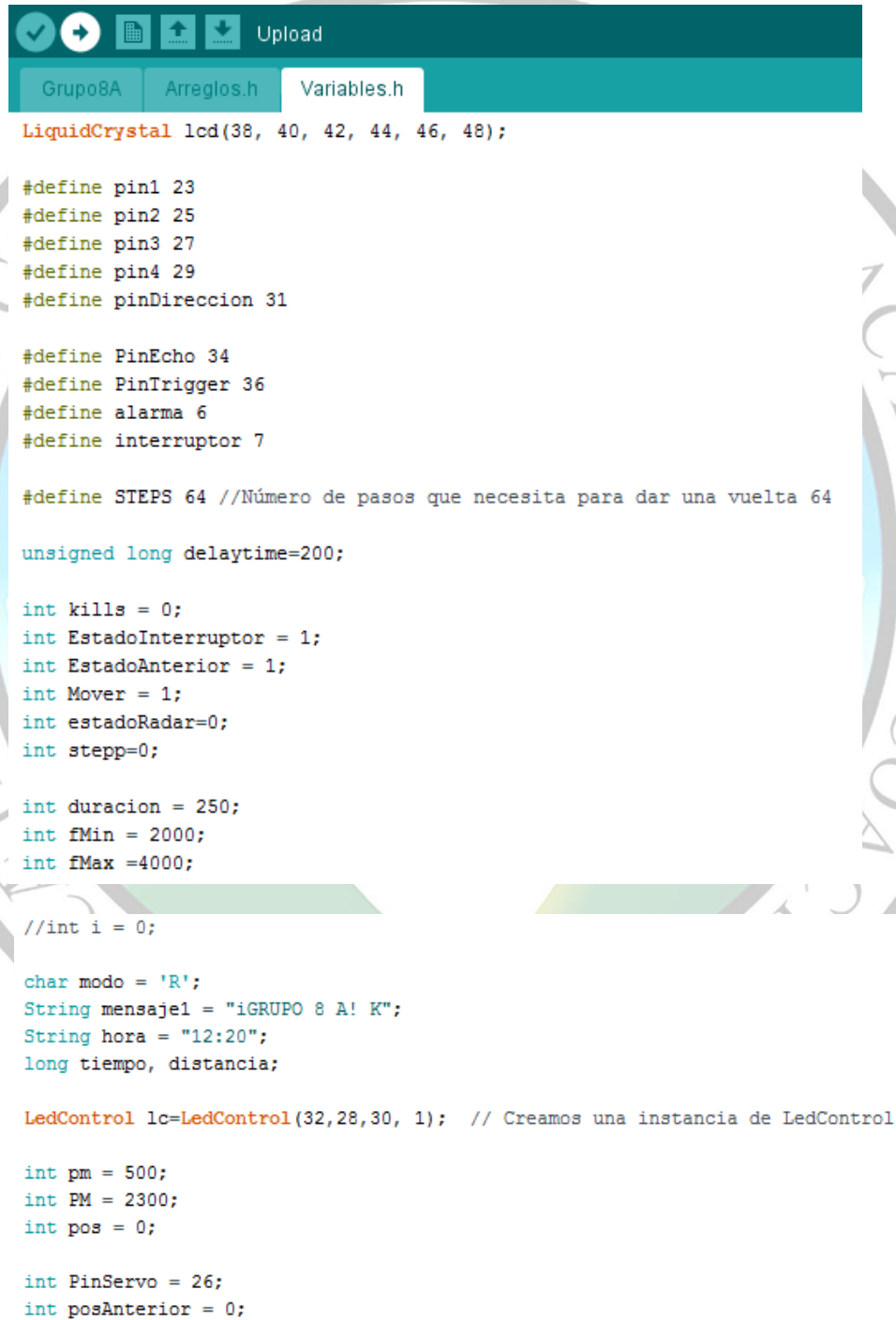
YOSELIN ANNELICE LEMUS LÓPEZ

201403819

EXPLICACION DE CODIGO:

Generalidades:

Se hizo uso de una cabecera llamada "Variables.h", la cual contiene todos los pines definidos para cada uno de los componentes que se utilizaron durante la elaboración de la práctica, así como también las variables globales que se utilizaron dentro del código.



```
LiquidCrystal lcd(38, 40, 42, 44, 46, 48);

#define pin1 23
#define pin2 25
#define pin3 27
#define pin4 29
#define pinDireccion 31

#define PinEcho 34
#define PinTrigger 36
#define alarma 6
#define interruptor 7

#define STEPS 64 //Número de pasos que necesita para dar una vuelta 64

unsigned long delaytime=200;

int kills = 0;
int EstadoInterruptor = 1;
int EstadoAnterior = 1;
int Mover = 1;
int estadoRadar=0;
int stepp=0;

int duracion = 250;
int fMin = 2000;
int fMax =4000;

//int i = 0;

char modo = 'R';
String mensaje1 = "iGRUPO 8 A! K";
String hora = "12:20";
long tiempo, distancia;

LedControl lc=LedControl(32,28,30, 1); // Creamos una instancia de LedControl

int pm = 500;
int PM = 2300;
int pos = 0;

int PinServo = 26;
int posAnterior = 0;
```

Además, en **Setup** se configuraron los pines

```
void setup() {  
  
    pinMode(pin1, OUTPUT);  
    pinMode(pin2, OUTPUT);  
    pinMode(pin3, OUTPUT);  
    pinMode(pin4, OUTPUT);  
    pinMode(pinDireccion, INPUT);  
  
    stepper.setSpeed(200); |  
  
    Serial.begin (9600);  
    lcd.begin(16, 2);  
    pinMode(PinEcho, INPUT);  
    pinMode(PinTrigger, OUTPUT);  
    pinMode(alarma, 1);  
    pinMode(interruptor, OUTPUT);  
  
    lc.shutdown(0, false); // Activar las matrices  
    lc.setIntensity(0, 8); // Poner el brillo a un valor intermedio  
    lc.clearDisplay(0); // Y borrar todo  
}
```

Y en **loop** se llaman las funciones principales, las cuales llaman y administran el funcionamiento de los demás componentes.

```
void loop() {  
    MostrarLCD();  
    ModoReconocimiento();  
}
```

Servo Motor:

Librería utilizada para poder controlar un motor servo fue <Servo.h> quien se encarga de manejar los ángulos envía por el usuario, las variables pm y PM se utilizaron para fijar el ancho mínimo y máximo pulso del servo, de debe de crear una variable de tipo servo que servirá para los distintos métodos utilizados por la librería, la variable posAnterior se utiliza para verificar la posición en la que se encuentra el servo motor y hacer el cambio a la nueva posición.

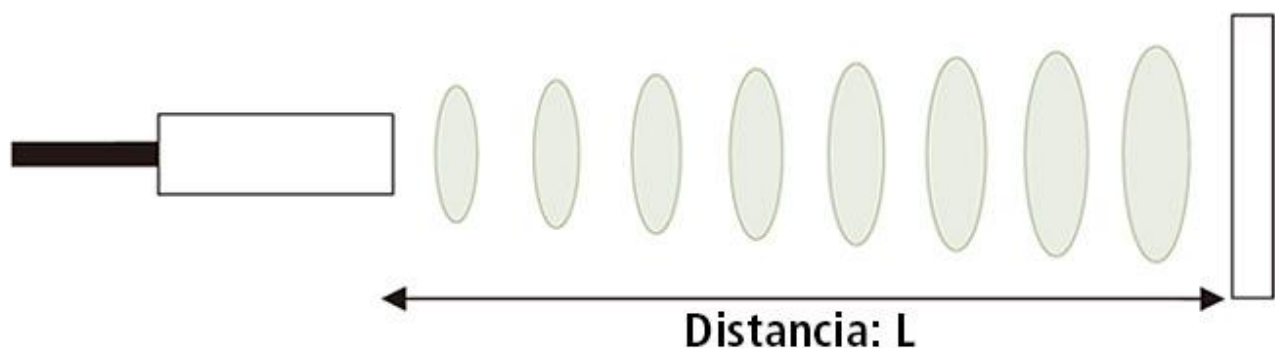
```
#include <Servo.h>  
  
int pm = 500;  
int PM = 2300;  
int pos = 0;  
  
Servo serv01;  
int PinServo = 26;  
int posAnterior = 0;
```

El método motorServo(<Parametros>) se utiliza para poder controlar los motores, verificando la posición en la que se encuentra a la posición a la cual quiere moverse, definiendo el tiempo al cual se debe de mover con un delay(), los parámetros utilizados, uno es para la posición en grados a la cual se debe de hacer el cambio y el segundo parámetro es para saber que motor se debe de mover indicando el pin al cual esta conectado al Arduino.

```
void motorServo(int posGrados, int PinServo){
    servol.attach(PinServo, pm, PM);
    if(posGrados <= 180){
        if(posGrados>posAnterior){
            for(int i = posAnterior; i<=posGrados; i++){
                servol.write(i);
                delay(30);
            }
            posAnterior = posGrados;
        }else if(posGrados<posAnterior){
            for(int i = posAnterior; i>= posGrados; i--){
                servol.write(i);
                delay(30);
            }
            posAnterior = posGrados;
        }else if(posGrados==posAnterior){
        }
    }
    servol.detach();
}
```

Sensor Ultrasónico:

Como su nombre lo indica, los sensores ultrasónicos miden la distancia mediante el uso de ondas ultrasónicas. El cabezal emite una onda ultrasónica y recibe la onda reflejada que retorna desde el objeto. Los sensores ultrasónicos miden la distancia al objeto contando el tiempo entre la emisión y la recepción.



```

void SensorUltrasonico()
{
  PulsoTrigger();
  if (distancia >= 500 || distancia <= 0)
  { // si la distancia es mayor a 500cm o menor a 0cm
    Serial.println("Distancia mayor a 500 cm o menor a 0 cm"); // no mide nada
  }
  else
  {
    Serial.print(distancia); // envia el valor de la distancia por el puerto serial
    Serial.println("cm"); // le coloca a la distancia los centímetros "cm"
    digitalWrite(alarma, 0); // en bajo el pin alarma
  }
  if (distancia <= 50 && distancia >= 10)
  {
    Serial.println("Alarma.....");
    ALARMA();
    ModoDeteccion();
  }
}

```

Con el siguiente método se llama a un pulso al pin de Trigger que es el que lo activa para poder mandar las ondas. El cual cambia el valor de la variable distancia que es global y con el cual se compara abajo para ver entre que rango se encuentra la distancia, si la distancia está entre el rango de 50 y 10 entonces se activa la alarma y entra al modo deteccion que es donde se procede a calcular el angulo respecto a la distancia y se dispara al objetivo.

Con respecto al pulso del trigger este recibe un pulso de 10 microsegundos el cual le permite activarlo. Y luego uno de 2 microsegundos en apagado. Luego de eso se calcula el tiempo el que se tardó en regresar la señal por el pin Echo para poder determinar el tiempo, luego se aplica la fórmula para calcular la distancia a la cual el objetivo hizo que las ondas ultrasónicas rebotaran. (donde distancia es una variable global por lo que no hay necesidad de estar pasando esa variable entre los parámetros de las funciones.

```

void PulsoTrigger()
{
  digitalWrite(PinTrigger, LOW);
  delayMicroseconds(2);
  digitalWrite(PinTrigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(PinTrigger, LOW);
  tiempo = pulseIn(PinEcho, HIGH);
  distancia = (tiempo/2) / 29;
  if(distancia>500){distancia = 500;}
}

```

Pantalla LCD (16 x 2):

La lcd funciona con la librería LiquidCrystal, se crea una variable lcd la cual nos permitirá manejar todas las funciones de la lcd, con el método .setCursor(,#) se especifica en que posición empezará a escribirse y la línea en la que empezará a escribirse de la lcd.

```
void MostrarLCD()
{
  lcd.setCursor(0,0);
  lcd.print(mensaje1+String(kills));
  lcd.setCursor(0,1);
  switch(modos)
  {
    case 'R':
      lcd.print("H"+hora+ " "+String(modos));
      lcd.print(" ");
      lcd.display();
      break;
    case 'D':
      lcd.print("H"+hora+ " "+String(modos)+" "+distancia+ " cm");
      lcd.display();
      break;
  }
}
```


Motor stepper

Para el movimiento del motor stepper se utilizó la librería **<Stepper.h>**, la cual permite manejar el motor de una manera sencilla.

En esta línea únicamente se define una constante, la cual representa el número de pasos que el motor necesita para que alcance un giro de 5.625 grados

```
#define STEPS 64 //Número de pasos que necesita para dar una vuelta 64
```

Se crea una variable de tipo **Stepper**, a través de la cual se realizan las acciones para el motor. El primer parámetro es la constante que se declaró anteriormente, y seguido de ella vienen los cuatro pines para el manejo del motor (pin1, pin2, pin3, pin4)

```
Stepper stepper(STEPS, pin1,pin3,pin2,pin4);
```

Se define el número de revoluciones por minuto del motor

```
stepper.setSpeed(200);
```

El siguiente método verifica el estado del interruptor de dirección del motor, en caso de que la variable **EstadoInterruptor** tuviera un valor de 1 se mueve el reloj en dirección de las agujas del reloj y en cualquier otro caso se mueve el motor en dirección contraria a las agujas del reloj.

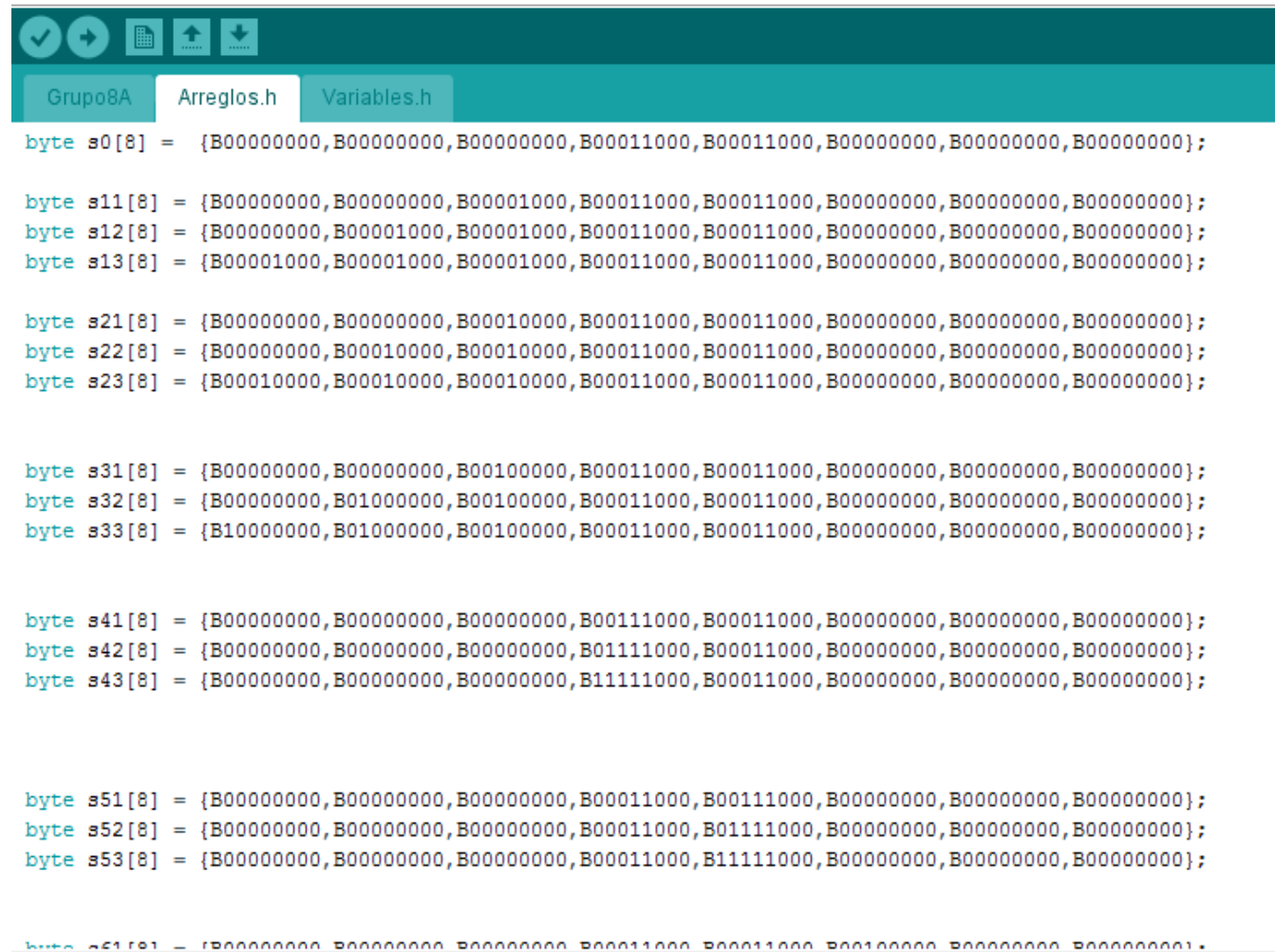
```
void MoverStepper()
{
    if(EstadoInterruptor==1)
    { //MOVER AGUJAS DEL RELOJ
        stepper.step(-128);
        stepp-=128;
        if(stepp<=0) {stepp=2048;}
    }
    else
    { //EN CONTRA DE LAS AGUJAS
        stepper.step(128);
        stepp+=128;
        if(stepp>=2048) {stepp=0;}
    }
}
```

Para el manejo de la hora se utilizó la función **millis()**, la cual devuelve el tiempo que el Arduino ha estado en ejecución, lo único que se hace es dividir lo que devuelve la función dentro de 1000 para obtener el tiempo en segundos, luego se verifica si el tiempo es mayor a 59 para detectar si ya pasó más de un minuto y así poder sumarle un minuto al tiempo actual y si el tiempo no es mayor que 59 la variable t2 permanece con el mismo valor.

```
void MostrarLCD()
{
    t1=millis()/1000;
    if(t1>59) {t1=t1/60;t2=minutos+t1;}
    else {t2=minutos;}
}
```

Matriz Led (8 x 8):

Para la mostrar el radar en la matriz de led se hizo uso de una cabecera llamada “Arreglos.h” donde se encontraban el conjunto de posibles estados o posiciones que la matriz pudiera tener para formar así el radar que indica la ubicación del sensor ultrasónico.



```
byte s0[8] = {B00000000,B00000000,B00000000,B00011000,B00011000,B00000000,B00000000,B00000000};

byte s11[8] = {B00000000,B00000000,B00001000,B00011000,B00011000,B00000000,B00000000,B00000000};
byte s12[8] = {B00000000,B00001000,B00001000,B00011000,B00011000,B00000000,B00000000,B00000000};
byte s13[8] = {B00001000,B00001000,B00001000,B00011000,B00011000,B00000000,B00000000,B00000000};

byte s21[8] = {B00000000,B00000000,B00010000,B00011000,B00011000,B00000000,B00000000,B00000000};
byte s22[8] = {B00000000,B00010000,B00010000,B00011000,B00011000,B00000000,B00000000,B00000000};
byte s23[8] = {B00010000,B00010000,B00010000,B00011000,B00011000,B00000000,B00000000,B00000000};

byte s31[8] = {B00000000,B00000000,B00100000,B00011000,B00011000,B00000000,B00000000,B00000000};
byte s32[8] = {B00000000,B01000000,B00100000,B00011000,B00011000,B00000000,B00000000,B00000000};
byte s33[8] = {B10000000,B01000000,B00100000,B00011000,B00011000,B00000000,B00000000,B00000000};

byte s41[8] = {B00000000,B00000000,B00000000,B00111000,B00011000,B00000000,B00000000,B00000000};
byte s42[8] = {B00000000,B00000000,B00000000,B01111000,B00011000,B00000000,B00000000,B00000000};
byte s43[8] = {B00000000,B00000000,B00000000,B11111000,B00011000,B00000000,B00000000,B00000000};

byte s51[8] = {B00000000,B00000000,B00000000,B00011000,B00011000,B00000000,B00000000,B00000000};
byte s52[8] = {B00000000,B00000000,B00000000,B00011000,B01111000,B00000000,B00000000,B00000000};
byte s53[8] = {B00000000,B00000000,B00000000,B00011000,B11111000,B00000000,B00000000,B00000000};

byte s51[8] = {B00000000,B00000000,B00000000,B00011000,B00011000,B00010000,B00000000,B00000000};
```

Y en el método llamado “ActualizarMatriz”, lo que se hace es mostrar en la matriz la posición en la que se encuentra el sensor, simulando así un Radar. Para lo cual hace uso de una variable llamada “contador” la cual indica los pasos que el motor Steper da para colocarse en los grados que se necesitan para formar dicho radar y así poder actualizar los conjuntos de arreglos para visualizarlos.


```

void ActualizarMatriz(int contador)
{
switch(contador) {
    case 0:
        for(int i=0;i<8;i++) {lc.setRow(0,i,s0[i]);}
        delay(delaytime);
        for(int i=0;i<8;i++) {lc.setRow(0,i,s11[i]);}
        delay(delaytime);
        for(int i=0;i<8;i++) {lc.setRow(0,i,s12[i]);}
        delay(delaytime);
        for(int i=0;i<8;i++) {lc.setRow(0,i,s13[i]);}
        delay(delaytime);
        break;
    case 128:
        for(int i=0;i<8;i++) {lc.setRow(0,i,s0[i]);}
        delay(delaytime);
        for(int i=0;i<8;i++) {lc.setRow(0,i,s21[i]);}
        delay(delaytime);
        for(int i=0;i<8;i++) {lc.setRow(0,i,s22[i]);}
        delay(delaytime);
        for(int i=0;i<8;i++) {lc.setRow(0,i,s23[i]);}
        delay(delaytime);
        break;
}
}

```

Alarma:

Cuando el sensor detecta un intruso u objetivo suena una alarma indicando que detecto un blanco al cual dispararle. Para esto se hace uso del método “tone” para que suene la alarma.

```

void ALARMA()
{
    for(int i=fMin; i<=fMax; i++){
        tone(alarma,i,duracion);
    }
    for(int i=fMax; i>=fMin; i--){
        tone(alarma,i,duracion);
    }
}

```

DIAGRAMA FINAL DE LA SOLUCION:

En el siguiente diagrama se le da referencia a como está conectado el circuito dentro del encapsulado, en el cual la LED representa la alarma o Buzzer y el Switch representa al interruptor, también se muestran cómo van conectados los pines de salida del Arduino a la matriz de leds.

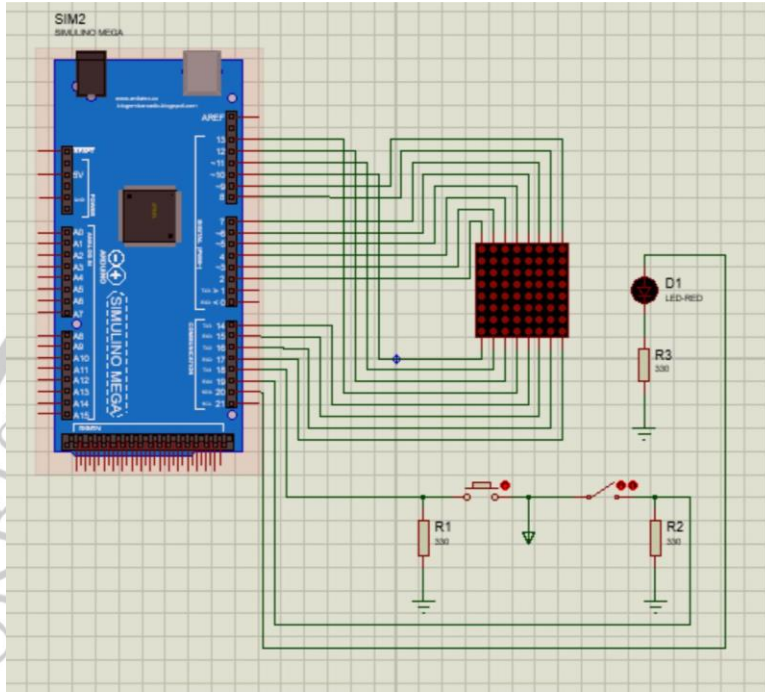
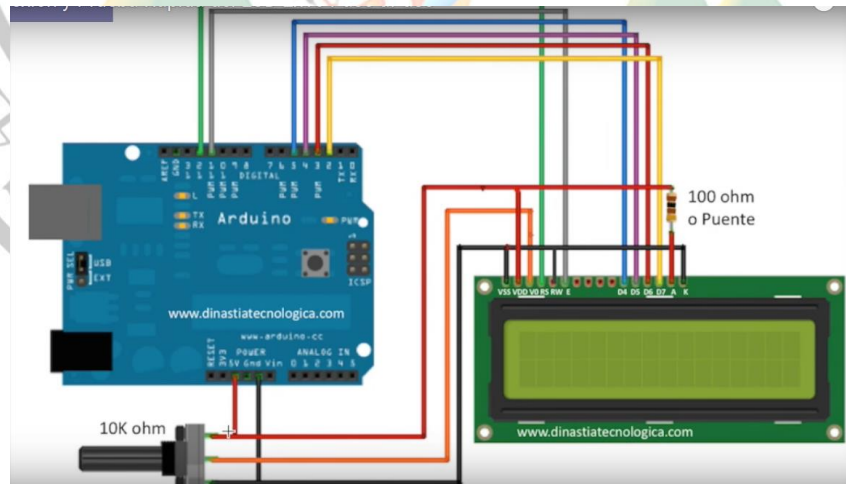
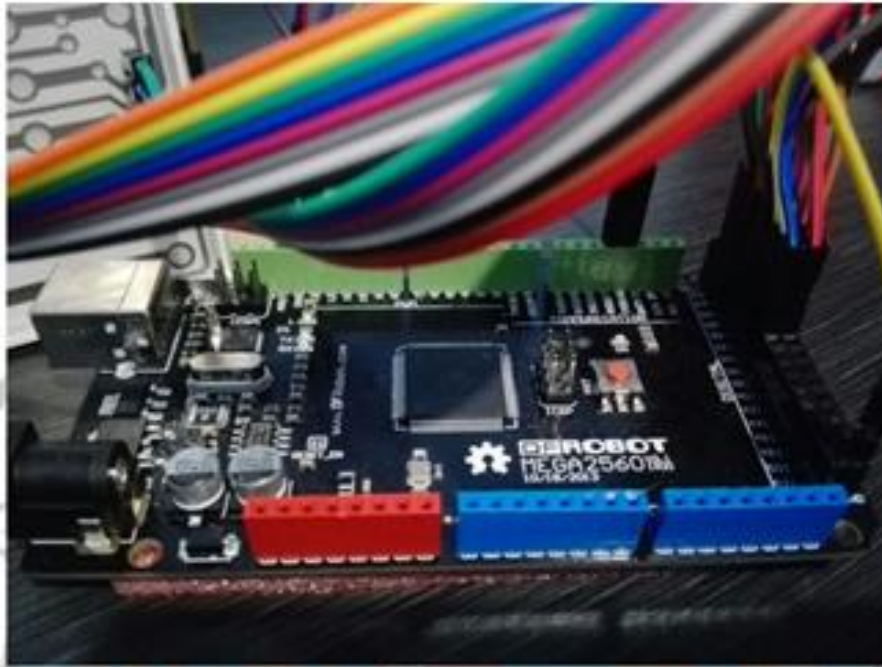


Diagrama de LCD

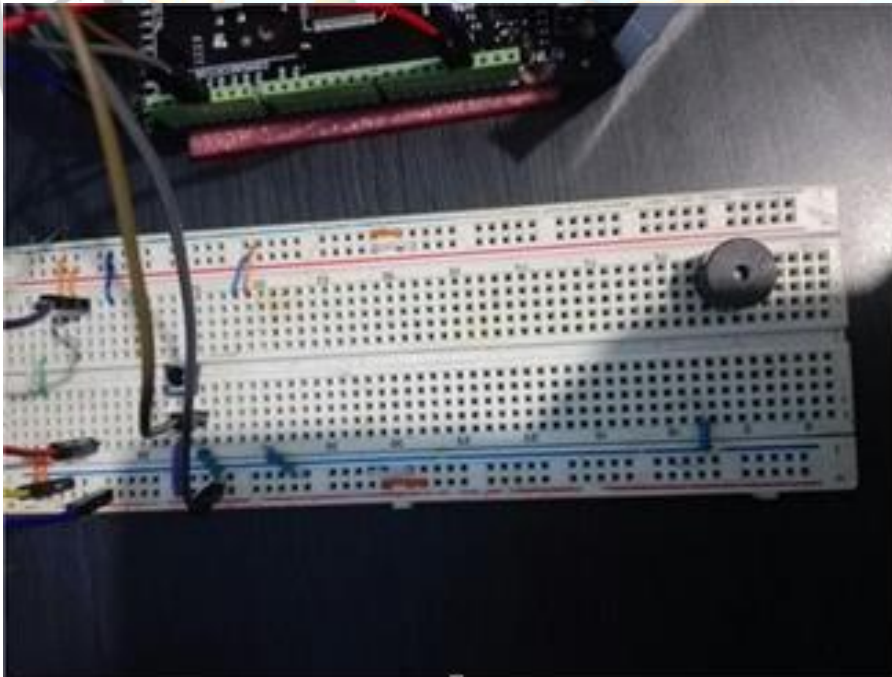


COMPONENTES UTILIZADOS:

ARDUINO MEGA:



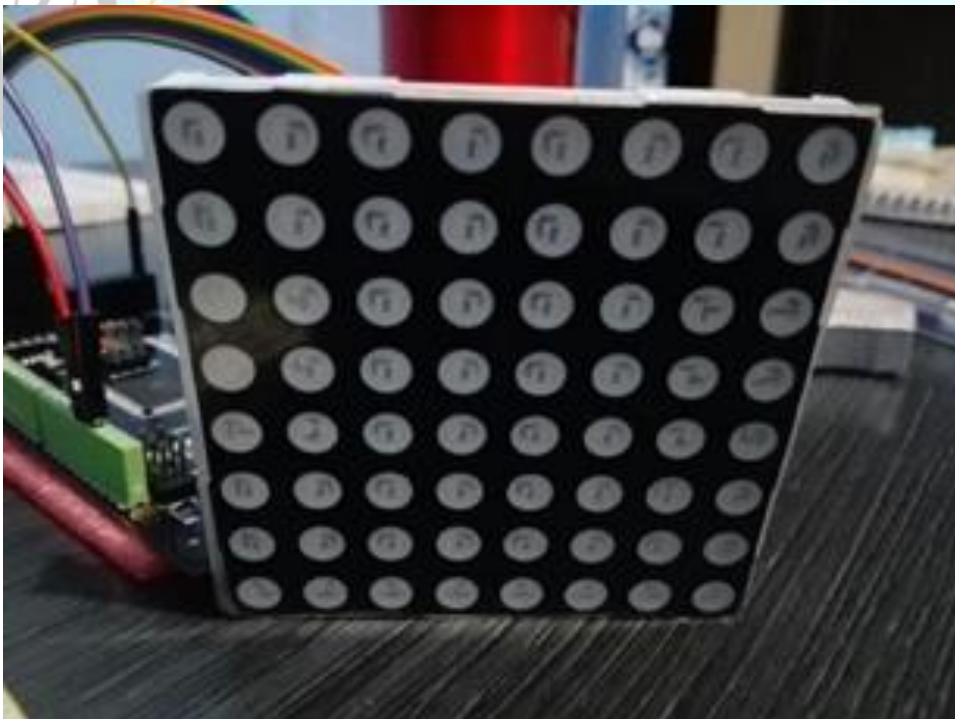
PROTOBOARD:



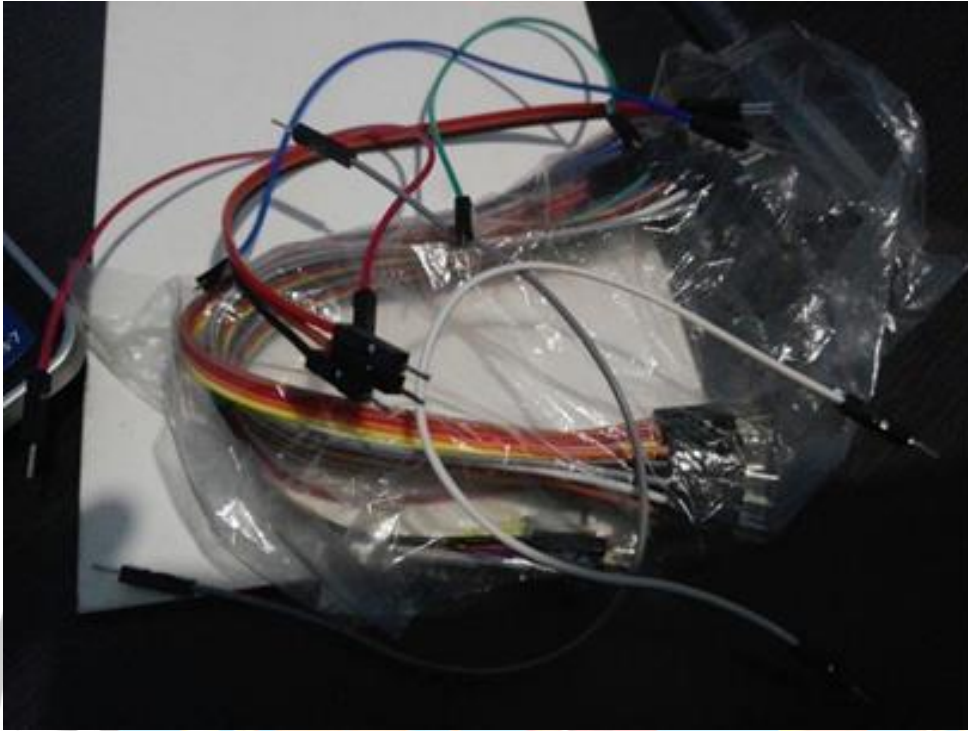
BUZZER E INTERRUPTOR:



MATRIZ DE LEDS DE 8X8:



JUMPERS MACHO-MACHO Y HEMBRA-MACHO:



SENSOR ULTRASONICO:

